DEEP LEARNING AUTOMATED WORKFLOW FOR CLOUD SEGMENTATION IN REMOTE SENSING IMAGES

Diego Henrique M. Matos[1], **Alber H. Sanchez**[2], **Tiago G. S. Carneiro** [3]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
– Belo Horizonte, MG – Brazil

[2]Instituto Nacional de Pesquisas Espaciais
– São José dos Campos, SP – Brazil

[3]Departamento de Computação –Universidade Federal de Ouro Preto
– Ouro Preto, MG – Brazil

diegohmm@ufmg.br, alber.ipia@inpe.br, tiago@ufop.edu.br

Abstract. *In this work, we propose an open source and automated workflow for semantic segmentation of remote sensing images. Even though it can be used in other sensors, to evaluate this workflow, a case study has been conducted applying a deep learning algorithm for segmenting clouds in images from WFI sensor, onboard CBERS-4A satellite. Since WFI does not have a tailor-made cloud segmentation algorithm, we customized our workflow based on the U-net neural network to fulfill this gap. Our results are promising according to our tests, although some problems were identified, like false positives over high albedo targets. These problems suggest improvements that could be tackled in the future.*

1. Introduction

A step to quantify and identify deforestation in remote sensing images is identifying clouds. On average, the cloud cover on Earth's surface is between 58% [Rossow and Schiffer 1999] and 66% [Zhang et al. 2004]. When the object of analysis of a remote sensing image is on Earth surface, like forests and cities, the clouds become obstacles between the remote sensor and the object. Thus, detecting and removing clouds are essential steps in remote sensing image analysis [Zhu and Woodcock 2012].

When monitoring large regions, handling data for segmenting clouds is a challenge. This is due to the great number of scenes that need to be processed to form a mosaic covering a region. Each scene is composed by multispectral images, each one containing data from specific spectral bands. Thus, a unique scene implies processing one gigabyte of data. Additionally, each scene has metadata that must be preserved during segmentation process.

In this context, this work develops a deep learning cloud segmentation workflow, which can run on personal computers due to its reduced computational requirements. To accomplish this, scenes are cropped in batches and loaded into memory at the training step, preserving its spatial metadata. This workflow can deal with images from several remote sensors with different characteristics. The only requirement is that input data should be provided in the GeoTIFF file format. Thus, the problem solved in this study is to automatically handle geolocalized images for performing semantic segmentation tasks, allowing the development of novel machine learning-based cloud segmentation algorithms.

The semantic segmentation problem is defined as follows: given an image I, for each pixel X present in the image, determine to which class the pixel belongs. Where i is the image row index, N is the image height, j is the image column index, M is the image width, k is the image band index, and L is the number of the image's channels. Formally, let I be a remote sensing image so that I = $\{Xijk|i = 1, ...N, j = 1, ...M, k = 1, ...L\}$, the semantic segmentation problem is to determine whether the pixel Xij belongs to a determinate class.

In order to evaluate the workflow, this work also conducted a case study. A collection of results obtained from an U-net neural network architecture applied to CBERS-4A WFI remote images is discussed with the goal of evaluating the proposed workflow in cloud segmentation tasks. Images from CBERS-4A Wide Field Camera (WFI) sensor were chosen because they are designed to monitor Amazon rain forest [Souza et al. 2019] and track back deforestation and forest fires. CBERS-4A is capable of making revisits within five days to a certain area [Epiphanio 2011]. This feature makes it a good choice for deforestation monitoring tasks.

There are some systems to quantify and identify deforestation using WFI images, like PRODES and DETER [Souza et al. 2019]. However, even disclosing its data, CBERS-4A, a satellite designed by Brazil and China, does not have its own algorithm for cloud semantic segmentation. Currently, INPE uses an algorithm called Cmask [Qiu et al. 2020] to segment clouds [Ferreira et al. 2020]. Nevertheless, Cmask has been originally developed for processing Landsat images and it is not implemented to any remote sensor onboard CBERS-4A. Because of that, the algorithm underperforms. This fact may imply that many images are useless in the processes of forest monitoring.

The rest of this work is organized as follows. Section 2 describes two different branches of the state-of-the-art semantic cloud segmentation and related work. Section 3 presents the methodology, including each step of the proposed workflow, justifying the way they are prepared. Section 4 shows the results of the presented tool applied to a cloud segmentation task. Finally, section 5 presents the conclusion and future work to complement the tool.

2. Related Work

Cloud segmentation could be divided into two large categories: physical - based algorithms and machine learning algorithms. The physical based algorithms use rules based on physical properties, like, for example, the relative angle of sensors to clouds and the angle from the sun, to extract a potential cloud layer and a potential cloud shadow layer [Foga et al. 2017]. This approach also has some disadvantages. First, there are several different remote sensors, with their own spectral bands and resolutions. As a consequence, their physical characteristics are inherent to each sensor and the algorithms can rarely be generalized to more than one sensor. Second, each image goes through the entire algorithm whenever it is segmented, therefore, a physical algorithm, like Fmask [Zhu and Woodcock 2012], can take 0.5 to 6 min to segment a scene on an 8 core Linux server. This makes semantic segmentation a time-consuming task to perform.

Fmask is one of the most used algorithms to segment clouds in Landsat TM images [Zhu and Woodcock 2012]. Currently, it is common to use it for benchmarking other algorithms. It is employed, for example, to evaluate another implementation of Fmask, called CFMask, comparing its performance with ”Automated Cloud Cover Assessment” (ACCA) system [Foga et al. 2017].

The Sen2cor algorithm also is a physics-based algorithm, that can classify clouds using just physical data presented in each spectral band [Louis et al. 2016]. This algorithm is created for Sentinel-2 Level-1C products. Cmask, Fmask, and sen2cor algorithms provide evidences on the relationship between physical based algorithms and remote sensor, in a way that the major part of algorithms difficultly adapt to different sensors.

A neural network architecture has been developed allowing the input of remote sensing images of different dimensions, the model is called CloudFCN [Francis et al. 2019]. The mechanism of receive different input images is very useful to allow entering data from more than one remote sensor, thus making the model more generic. Using the Biome dataset [Foga et al. 2017] the model showed an accuracy of 82.81% for RGB scenes and 91.00% with all spectral information versus CFmask [Foga et al. 2017] with 65.69% using all spectral bands available. The lack of spectral bands, using only RGB bands, made it difficult to the model to predict some cases of snow, sand (when it is used remote sensor of high resolution) and other objects that present high albedo.

A new approach for detecting clouds even with hardly distinguishable scenery has been created [Jeppesen et al. 2019]. The model is a neural network built from U-Net and got meaningful results. The ”Biome” and ”Sparcs” datasets [Foga et al. 2017] were used to evaluate the model divided it into three situations. First, the model is trained with the Biome dataset and evaluated with Sparcs, and vice versa. Second, the Fmask algorithm generates the groundtruth masks. Finally, the model is trained and tested on the same dataset using image-based cross-validation, where the training data are the groundtruth cloud masks. The model improves the FMask even when using just RGB bands in all the three cited cases of evaluation. This case study has shown that, even with images extracted from a remote sensor with few spectral bands, like CBERS-4A WFI, a deep learning model can produce trustworthy cloud masks.

Currently, the algorithms for semantic segmentation on remote sensing images mainly use deep learning strategies [Yuan et al. 2021]. This occurs because in the last decade, more and more deep learning algorithms have demonstrated good performance in tasks as object classification, performing better than humans in some cases [Lee et al. 2017].

3. Workflows for Cloud Segmentation with Deep Learning Algorithms

The workflow is divided into two main components: model calibration and production model. Workflow components are decoupled from each other. These steps are called in two Jupyter notebook files [Kluyver et al. 2016]. Each notebook implements the main control flow of each component, one for model calibration and another for the production model. The notebook files act as an interface for the user, in which the user may script the workflow in Python programming language [vanRossum 1995], configuring, and connecting steps.

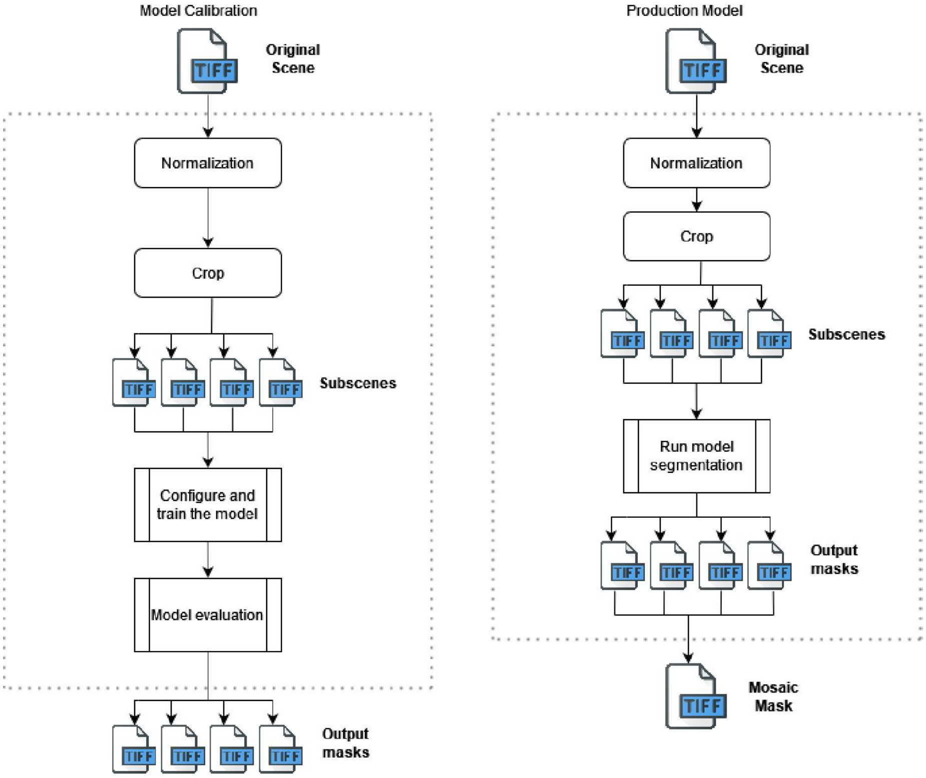

Figure 1. Schema of the proposed workflow. The model calibration (left) is meant to pre-processing the dataset and train the model. The production model (right) is meant to apply new images to be segmented.

The model calibration workflow component consists of several steps: preprocessing remote sensing images (normalization and crop), configuring and training a deep learning model, and compute metrics to evaluate the trained model.

Each step is implemented as a Python function based on the Keras [Ketkar 2017] library that provides features for deep learning model development. The GDAL [Warmerdam 2008] and NumPy [Oliphant 2006] libraries have been used to handle remote sensing images without losing the georeferencing and ”no data values” metadata.

3.1. Dataset

Clouds are particularly different entities from objects traditionally segmented in deep learning tasks: the center of a cloud can be dense, with high albedo. However, the border of a cloud can be fuzzy, and can be confused with the background [Dowling and Radke 1990].

In this work, the Cmask algorithm is employed to produce a dataset of segmented images. This strategy allow us to focus on the workflow development. The current dataset is composed by 4 images from WFI remote sensor, that are divided into smaller images, as shown in the figure 2, of 256x256x4 pixels, totaling more than 10.760 images with its respective masks. Since the masks are obtained by applying the Cmask algorithm [Qiu et al. 2020] to each image, they have errors associated and do not correctly represent the ground-truth.

The first step in handling the dataset is to build a directory hierarchy that is divided into the following:

- Original scene folder, to be filled with the original scenes (with all bands);
- normalized scene folder, to be filled with the respective normalized scenes;
- Cropped production folder, to be filled with subscenes from the normalized scenes;
- Cropped production mask folder, to be filled with output masks estimated by the model for each subscene.

In sequence, each scene goes through a Min-Max normalization step, which uses a linear operation to transform the pixel’s values [Saranya and Manikandan 2013]. Here, the original pixel’s values ranging from 0 to 10.000 are mapped to new values between 0 and 1. The purpose of the normalization step is to transform the image’s pixels in a way that each image presents a similar distribution of its pixel values, which helps the neural network optimize its parameters.

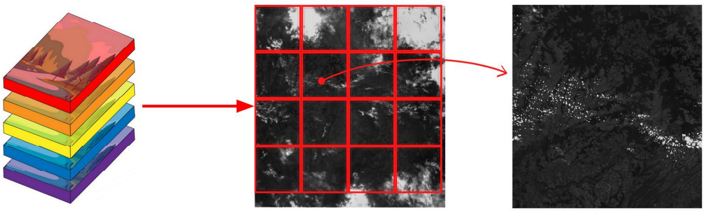

Figure 2. Demonstration of the image cropping step. On the left is the raw scene, in the center is the raw scene with the subscenes in red, on the right is the resulting subscene.

3.2. Deep Learning Model

The chosen neural network has an U-net architecture [Ronneberger et al. 2015], due to its performance and presence in the state-of-the-art. Its architecture is built in two parts. The former is called encoder, and the latter a decoder. The encoder step consists of applying convolutional operations followed by a maxpool downsampling to encode the input image into feature representations at multiple different levels. The decoder step consists of reconstructing the image through upsampling and concatenation at each layer level using the features learned in the encoder step. The enconder-decoder strategy is what makes U-nets good options for the segmentation task, extracting and mapping characteristics from an image.

The tensor representing the subscene can be divided into several dimensions depending on the available memory and the number of bands of the remote sensor. In this study case, each tensor is an image with (256 pixels x 256 pixels x 4 bands). We have chosen a U-net implementation from GitHub with configurable input dimensions [Zak 2021], so that, the resulting workflow can also be configured.

The Keras library contains many metrics available to interpret the confidence of a neural network output, except the F1-Score metric, so it is implemented. It is expected that the user defines a loss function to finish the neural network model. All the loss functions implemented in Keras library are allowed to be used as inputs at this step.

3.3. Training and Test Step

In the learning stage, the model must receive subscenes as input. The set of subscenes are called batches. To load the batches and not use all the machine's memory, a mechanism called "generator" is built to iterate among the images, loading a batch and, after training, reallocating memory to a new batch. In this way, the generator expects the batch size hyperparameter as the input.

After the model consumes the input images and output their respective estimated cloud masks, the ground truth masks will be compared to them. The metrics configured in the previous step are then computed and stored in each pair estimated-ground truth masks.

3.4. Production Model

The first step in handling the dataset is to build a directory hierarchy. The directory folders are divided into the following:

- Production scene folder, to be filled with the original scenes;
- Normalized scene folder, to be filled with the respective normalized scenes;
- Cropped production folder, to be filled with subscenes from the normalized scenes;
- Cropped production mask folder, to be filled with output masks estimated by the model for each subscene;
- Segmented mask folder, to be filled with the entire reconstructed mask.

If there are several scenes in the input dataset, the production model will create an entire directory hierarchy for each scene.

The next step after building the directory hierarchy is to normalize and crop the scenes, a process that is identical to the step in the Model Calibration component.

In the next step, the subscenes are inputted into the pre-trained neural network and the outputs are collected to build a mosaic using the ”WARP” function present on GDAL library.

4. Results

The workflow is entirely developed in the Python language and it is executed a 2.60GHz Intel Core i7-9750H machine, with 8 gigabytes of RAM in the Windows 10 operating system. The main employed libraries are Keras, for model implementation; GDAL, to manipulate the georeferenced images; and NumPy, for handle tensors. The following hyperparameters are used:

Table 1. Case study hyperparameter table

Hyperparameter	
Dimension	256
Bands	4
Loss Function	Binary cross-entrophy
Optmizer	Adam
Metric	F1-score
Patience	5
Epochs	20

In the case study conducted to evaluate the workflow, it is used a dataset composed by images from CBERS-4A WFI. These images have four spectral bands, Red, Blue, Green and near infra-red. Each scene presented in the training step, must have a ground-truth mask. Cmask algorithm extracted the masks used in this case study [Foga et al. 2017].

In the case study, each subscene is 256x256x4 pixels, being 256 pixels in width and height and 4 layers for red, green, blue, and near infrared spectral bands. The generator mechanism allows the pipeline to work using several images up to 2 gigabytes in size, even with a machine with only 8 gigabytes of RAM.

To choose the loss function parameter the scope of problem must be analyzed. Here, each existent pixel should be classified into two classes: ”cloud” or ”background.” This is a classic case of binary classification, where each pixel in an image either belongs or not to a class. To this scope of problem, the binary cross-entropy, which is based on Bernoulli’s formula, is suitable [Ruby and Yendapalli 2020].

The optimizer parameter is a function responsible for minimizing the loss [Géron 2019]. Adam optimizer is one of the most popular and famous gradient descent optimization algorithms [Bock and Weiß 2019]. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients, similar to momentum and the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both methods [Kingma and Ba 2014].

Finally, as the case study has no weight distinction for false positives and false negatives, we chose the F1-score metric to monitor the model learning.

4.1. Performance metrics

The first information generated by the workflow is the history graph represented on the Figure 3.

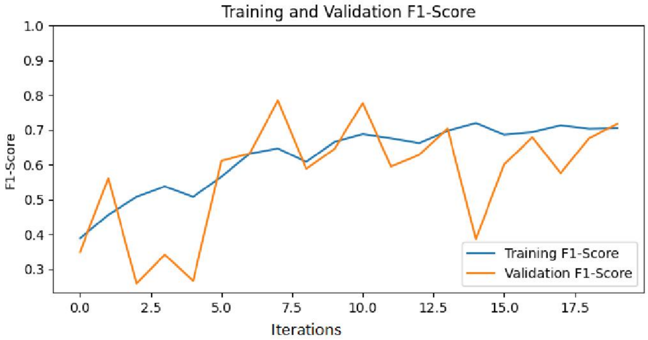

Figure 3. Evolution of the model along the training step.

With this image, it is possible to see metrics in each interaction during the training step, with the F1-score and loss being represented in each epoch. It is possible to observe that, during the training, the model improves at each interaction (Training F1-score) and that it shows a convergence after fifteen interactions. However, not always the validation metrics at the training phase confirm the model performance.

Below, Figure 4 shows some masks from the tested subscenes. The input subscenes appear in the left column, the input masks in the middle column, and the predicted mask in the right column.

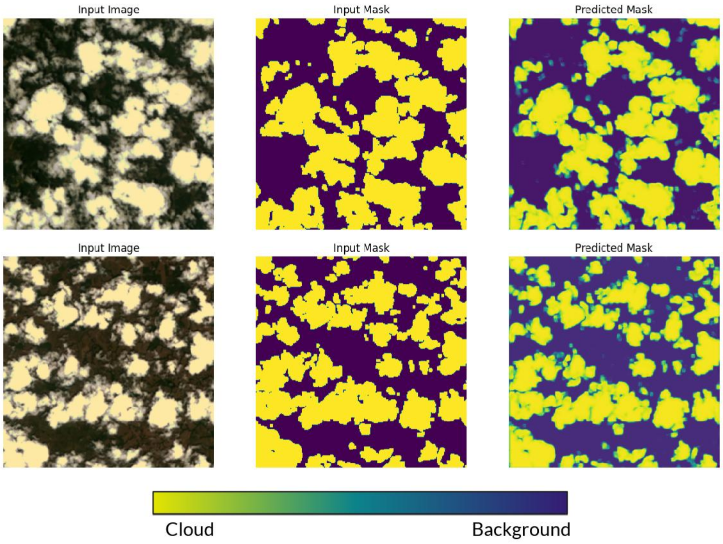

Figure 4. Some input data with the respective predicted masks for the U-net model. Yellow means a high level of probability that the pixel is a cloud. Blue means that there is a low probability that the pixel is a cloud.

The resulting masks present a color gradient between the background color (purple) and the cloud color (yellow), according to the confidence level of the model in the classification. To make a binary mask, another hyperparameter must be defined, a confidence-level threshold. If the threshold is set to an arbitrarily chosen value such as 0.97, only pixels with 97% confidence will be colored in yellow as clouds. Everything else will be purple. Therefore, it is important to find a threshold value that best avoids wrong coloring. We avoid arbitrarily choosing a threshold value in our case study, preserving the level of confidence analysis.

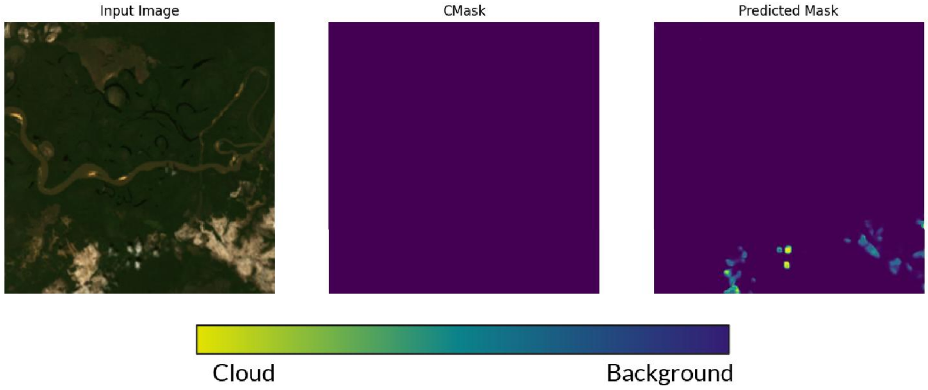

Figure 5. Some input data with the respective predicted mask that present a case of false positive cloudy pixels. Yellow means a high level of probability that the pixel is a cloud. Blue means that there is a low probability that the pixel is a cloud.

As shown in Figure 5, exposed soil has a high albedo compared with vegetation, causing the algorithm to missclassify.

The next step is to assess how the workflow works in a production scenario. After entering the scene in the workflow, the entire scene is segmented and its output mask is reconstructed and saved in the ”Merged Files” directory.

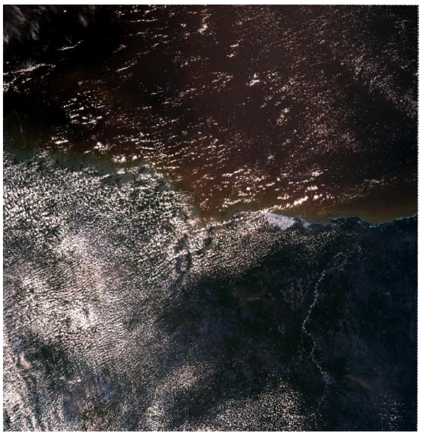

Figure 6. Raw Scene.

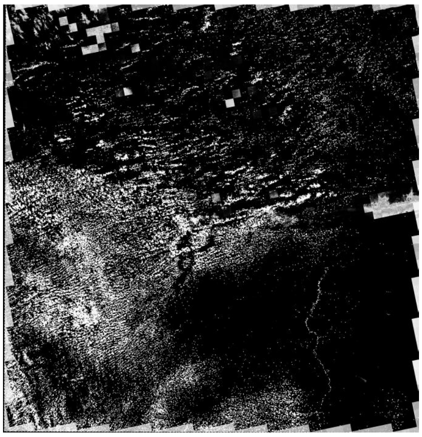

Figure 7. Mosaic mask.

Figure 7 shows a raw image from CBERS-4A dataset on the left, which the deep learning model has never seen. On the right side, it shows the mosaic of the output-segmented masks. The mosaic image does not have a defined threshold for pixel confidence levels. In this way, the pixel values are a gradient of values between 0 and 1. By applying a threshold, it is possible to select only pixels where the model has been segmented with high confidence. In Figure 8 no threshold is set, and the Figure 9 a 97% confidence limit is used.

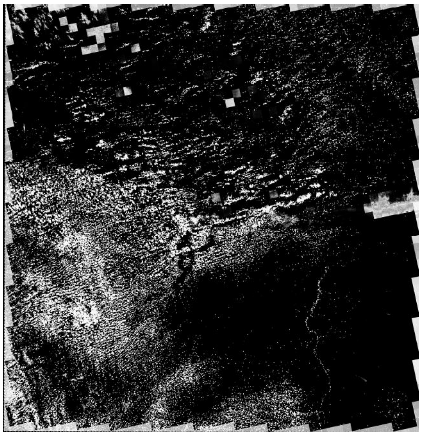

Figure 8. Mosaic mask.

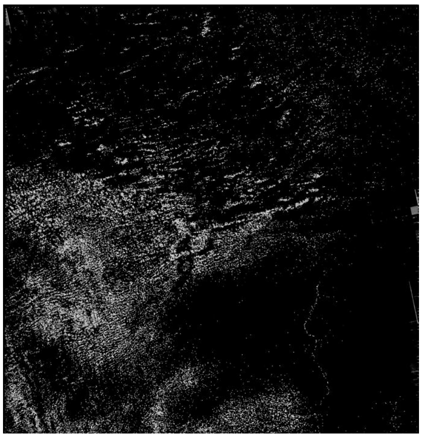

Figure 9. Mosaic mask with threshold.

The two last Figures show the importance of defining a threshold for classifying pixels. This process is called threshold tuning and it requires balancing the recall-precision trade-off [Buckland and Gey 1994]. There are strategies for defining a threshold in segmentation masks, but they are not addressed in this study.

5. Conclusion

This work developed and evaluated a workflow for semantic segmentation of remote sensing images. The proposed workflow handles the dataset automatically, requiring only hyperparameter inputs. The workflow is distributed as a free and open source software, under the LGPL GNU license. Its source code can be found in the repository: https://gitlab.com/ufopterralab/projeto-segmentacao-semanticas-cbers-4a-wfi.

The workflow can be configured for a more powerful machine (increasing sub-scene size and batch size) or running on a personal computer, with less computing power. Even with only 8 gigabytes of RAM, the workflow can train a deep learning model with several images of more than 2 gigabytes each one. The choice of a threshold is excluded from scope of this project, but is left open for future work.

The initial idea of building this workflow was to improve semantic segmentation for the CBERS-4A remote sensor, but in the process of building the dataset and workflow, it became clear that the workflow itself could be a project that would help scientists working with semantic segmentation of remote sensing images, requiring only the hyper-parameterization of the model to obtain an initial result.

In the process of building the case study dataset, we identified a scientific and technological gap in cloud removal algorithms for the CBERS-4A WFI remote sensor. Semantic segmentation algorithms that use deep learning appear able to outperform Cmask, which is currently used by INPE. Additionally, a groundtruth dataset is missing for this remote sensor, limiting the attempts to implement and evaluate other machine learning models to segment could in CBERS-4A WFI images.

References

Bock, S. and Weiß, M. (2019). A proof of local convergence for the adam optimizer. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Buckland, M. and Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19.

Dowling, D. R. and Radke, L. F. (1990). A summary of the physical properties of cirrus clouds. *Journal of Applied Meteorology and Climatology*, 29(9):970–978.

Epiphanio, J. C. N. (2011). Cbers-3/4: características e potencialidades. In *Proceedings of the Brazilian Remote Sensing Symposium, Curitiba, Brazil*, volume 30, page 90099016.

Ferreira, K. R., Queiroz, G. R., Vinhas, L., Marujo, R. F., Simoes, R. E., Picoli, M. C., Camara, G., Cartaxo, R., Gomes, V. C., Santos, L. A., et al. (2020). Earth observation data cubes for brazil: Requirements, methodology and products. *Remote Sensing*, 12(24):4033.

Foga, S., Scaramuzza, P. L., Guo, S., Zhu, Z., Dilley Jr, R. D., Beckmann, T., Schmidt, G. L., Dwyer, J. L., Hughes, M. J., and Laue, B. (2017). Cloud detection algorithm comparison and validation for operational landsat data products. *Remote sensing of environment*, 194:379–390.

Francis, A., Sidiropoulos, P., and Muller, J.-P. (2019). Cloudfcn: Accurate and robust cloud detection for satellite imagery with deep learning. *Remote Sensing*, 11(19):2312.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Jeppesen, J. H., Jacobsen, R. H., Inceoglu, F., and Toftegaard, T. S. (2019). A cloud detection algorithm for satellite imagery based on deep learning. *Remote sensing of environment*, 229:247–259.

Ketkar, N. (2017). Introduction to keras. In *Deep learning with Python*, pages 97–111. Springer.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In Loizides, F. and Schmidt, B., editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press.

Lee, J.-G., Jun, S., Cho, Y.-W., Lee, H., Kim, G. B., Seo, J. B., and Kim, N. (2017). Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584.

Louis, J., Debaecker, V., Pflug, B., Main-Knorn, M., Bieniarz, J., Mueller-Wilm, U., Cadau, E., and Gascon, F. (2016). Sentinel-2 sen2cor: L2a processor for users. In *Proceedings Living Planet Symposium 2016*, pages 1–8. Spacebooks Online.

Oliphant, T. E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Qiu, S., Zhu, Z., and Woodcock, C. E. (2020). Cirrus clouds that adversely affect landsat 8 images: what are they and how to detect them? *Remote Sensing of Environment*, 246:111884.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Rossow, W. B. and Schiffer, R. A. (1999). Advances in understanding clouds from isccp. *Bulletin of the American Meteorological Society*, 80(11):2261–2288.

Ruby, U. and Yendapalli, V. (2020). Binary cross entropy with deep learning technique for image classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(10).

Saranya, C. and Manikandan, G. (2013). A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET)*, 5(3):2701–2704.

Souza, A., Monteiro, A. M. V., Rennó, C. D., Almeida, C. A., Valeriano, D. d. M., Morelli, F., Vinhas, L., Maurano, L. E. P., Adami, M., Escada, M. I. S., et al. (2019). Metodologia utilizada nos projetos prodes e deter. *INPE: São José dos Campos, Brazil.*

vanRossum, G. (1995). Python reference manual. *Department of Computer Science [CS]*, (R 9525).

Warmerdam, F. (2008). The geospatial data abstraction library. In *Open source approaches in spatial data handling*, pages 87–104. Springer.

Yuan, X., Shi, J., and Gu, L. (2021). A review of deep learning methods for semantic segmentation of remote sensing imagery. *Expert Systems with Applications*, 169:114417.

Zak, K. (2021). keras-unet. `https://github.com/karolzak/keras-unet`.

Zhang, Y., Rossow, W. B., Lacis, A. A., Oinas, V., and Mishchenko, M. I. (2004). Calculation of radiative fluxes from the surface to top of atmosphere based on isccp and other global data sets: Refinements of the radiative transfer model and the input data. *Journal of Geophysical Research: Atmospheres*, 109(D19).

Zhu, Z. and Woodcock, C. E. (2012). Object-based cloud and cloud shadow detection in landsat imagery. *Remote sensing of environment*, 118:83–94.